

LAPORAN RESMI
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
INHERITANCE 2



Fadilah Fahrul Hardiansyah S.ST., M. Kom

Ratri Maria Manik

3121600039

D4 TEKNIK INFORMATIKA – B

PROGRAM STUDI TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
TA 2022/2023

A. TUGAS PENDAHULUAN

1. Ada berapa modifier untuk pengontrolan akses? Jelaskan masing – masing!
 - a. Public
Sebuah kelas, method, ataupun property yang mempunyai akses modifier public artinya bahwa kelas, method, ataupun property tersebut dapat diakses oleh kelas manapun
 - b. Protected
Sebuah method, ataupun property yang mempunyai akses modifier protected artinya bahwa method, ataupun property tersebut dapat diakses hanya oleh kelas turunannya (subclass) dan hanya dapat diakses oleh kelas yang satu package
 - c. Private
Access modifier private bersifat tertutup. Sesuai dengan konsep OOP Encapsulation, maka setiap variabel wajib untuk dilindungi hak aksesnya secara langsung dari luar. Oleh karena itu, variabel diberikan hak akses private dan untuk melakukan pengaksesan/perubahan data digunakan setter getter.
 - d. Default
Access modifier default bisa diakses oleh file di class yang sama atau package yang sama.
2. Apakah kegunaan kata kunci super? Jelaskan!
Kata kunci super dipakai untuk merujuk pada member dari parent class, sebagaimana kata kunci this yang dipakai untuk merujuk pada member dari class itu sendiri. Adapun format penulisannya yaitu
 - a. super.data_member → Data member pada parent class
 - b. super.function_member → Function member pada parent class
 - c. super() → konstruktor pada parent class
3. Apakah yang dimaksud dengan konstruktor tidak diwariskan?
konstruktorparent class tidak diwariskan ke sub class. Tetapi ketika suatu obyek anak dibuat dalam artian ketika konstruktor anak dijalankan maka konstruktor parent class dijalankan terlebih dahulu dan selanjutnya menyelesaikan konstruktor anak.

B. LATIHAN

1. Konstruktor tidak diwariskan

- Class Base

```
Base.java - Notepad
File Edit View

class Base{
    Base(int i){
        System.out.println("base constructor");
    }

    Base(){
    }
}
```

- Class Sup

```
Sup.java - Notepad
File Edit View

public class Sup extends Base{
    public static void main(String argv[]){
        Sup s= new Sup();
        s.derived(); //baris 1
    }

    Sup(){
        // baris 2
    }

    public void derived(){
        Base b = new Base(1); //baris 3
    }
}
```

- Hasil compile

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>javac Sup.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>java Sup
base constructor
```

2. Konstruktor tidak diwariskan

- Class Base

```
Base.java - Notepad
File Edit View

private class Base{
    Base(){
        int i = 100;
        System.out.println(i);
    }
}
```

- Class Pri

```
Pri.java - Notepad
File Edit View

public class Pri extends Base{
    static int i = 200;
    public static void main(String argv[]){
        Pri p = new Pri();
        System.out.println(i);
    }
}
```

- Hasil compile

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>javac pri.java
.\Base.java:1: error: modifier private not allowed here
private class Base{
    ^
1 error
```

Analisa :

Program tersebut terjadi error dikarenakan pada parent classnya(class Base) memiliki modifier private, sehingga tidak dapat diakses oleh class manapun, modifier private hanya dapat diakses oleh class yang sama.

3. Apa yang tampil di layar bila kode dibawah ini dijalankan?

- Class Y

```
Y.java - Notepad
File Edit View

class Y{
    Y(){
        System.out.print("Y");
    }
}
```

- Class X

```
X.java - Notepad
File Edit View

class X{
    Y b = new Y();
    X(){
        System.out.print("X");
    }
}
```

- Class Z

```
Z.java - Notepad
File Edit View

public class Z extends X{
    Y y = new Y();
    Z(){
        System.out.print("Z");
    }
    public static void main(String[] args){
        new Z();
    }
}
```

- Hasil compile

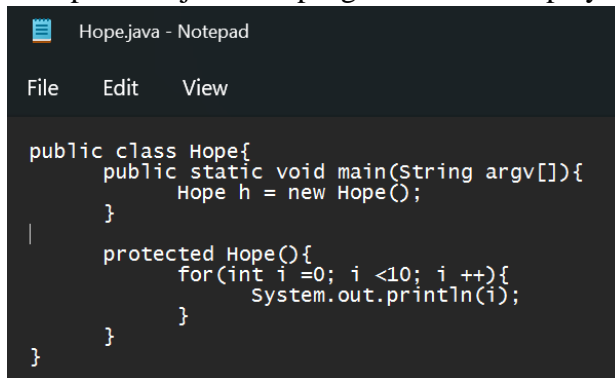
```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>javac Z.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>java Z
XYZ
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>_
```

Analisa :

Konstruktor yang pertama kali diproses yaitu Y, karena class Y merupakan konstruktor parent, sebelum subclass menjalankan konstruktornya sendiri, subclass akan menjalankan konstruktor parent terlebih dahulu (aturan tentang konstruktor tidak diwariskan). Selanjutnya akan menjalankan konstruktor X, dan terakhir menjalankan konstruktor Z.

4. Kompilasi dan jalankan program berikut! Apa yang terjadi? Jelaskan!



```
Hope.java - Notepad
File Edit View

public class Hope{
    public static void main(String argv[]){
        Hope h = new Hope();
    }

    protected Hope(){
        for(int i =0; i <10; i ++){
            System.out.println(i);
        }
    }
}
```

- Hasil compile

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>javac Hope.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>java Hope
0
1
2
3
4
5
6
7
8
9
```

Analisa :

Program tersebut digunakan untuk menampilkan bilangan dari 0 hingga 9. Dapat dilihat pada baris pertama memiliki modifier public, sedangkan pada bagian konstruktornya memiliki modifier protected.

C. TUGAS

1. Mengimplementasikan UML class diagram dalam program
 - a. Kendaraan.java

```
public class Kendaraan{
    protected int jmlRoda;
    protected String warna;

    public void setJmlRoda(int jmlRoda){
```

```

        this.jmlRoda = jmlRoda;
    }

    public int getJmlRoda(){
        return jmlRoda;
    }

    public void setWarna(String warna){
        this.warna = warna;
    }

    public String getWarna(){
        return warna;
    }
}

```

b. Mobil.java

```

public class Mobil extends Kendaraan{
    protected String bahanBakar;
    protected int kapasitasMesin;

    public void setBahanBakar(String bahanBakar){
        this.bahanBakar = bahanBakar;
    }

    public String getBahanBakar(){
        return bahanBakar;
    }

    public void setKapasitasMesin(int kapasitasMesin){
        this.kapasitasMesin = kapasitasMesin;
    }

    public int getKapasitasMesin(){
        return kapasitasMesin;
    }
}

```

c. Sepeda.java

```

public class Sepeda extends Kendaraan{
    protected int jmlSadel;
    protected int jmlGir;

    public void setJmlSadel(int jmlSadel){
        this.jmlSadel = jmlSadel;
    }

    public int getJmlSadel(){
        return jmlSadel;
    }

    public void setJmlGir(int jmlGir){
        this.jmlGir = jmlGir;
    }
}

```

```
    }

    public int getJmlGir(){
        return jmlGir;
    }
}
```

d. Truk.java

```
public class Truk extends Mobil{
    protected int muatanMaks;

    public void setMuatanMaks(int muatanMaks){
        this.muatanMaks = muatanMaks;
    }

    public int getMuatanMaks(){
        return muatanMaks;
    }
}
```

e. Taksi.java

```
public class Taksi extends Mobil{
    protected int tarifAwal;
    protected int tarifPerKm;

    public void setTarifAwal(int tarifAwal){
        this.tarifAwal = tarifAwal;
    }

    public int getTarifAwal(){
        return tarifAwal;
    }

    public void setTarifPerKm(int tarifPerKm){
        this.tarifPerKm = tarifPerKm;
    }

    public int getTarifPerKm(){
        return tarifPerKm;
    }
}
```

f. Tes.java

```
public class Tes{
    public static void main(String args[]){
        Truk truk1 = new Truk();
        truk1.setJmlRoda(4);
        truk1.setWarna("Kuning");
        truk1.setBahanBakar("Solar");
        truk1.setKapasitasMesin(1500);
        truk1.setMuatanMaks(1000);
    }
}
```

```
Truk truk2 = new Truk();
truk2.setJmlRoda(6);
truk2.setWarna("Merah");
truk2.setBahanBakar("Solar");
truk2.setKapasitasMesin(2000);
truk2.setMuatanMaks(5000);
```

```
Taksi taksi1 = new Taksi();
taksi1.setJmlRoda(4);
taksi1.setWarna("Orange");
taksi1.setBahanBakar("Bensin");
taksi1.setKapasitasMesin(1500);
taksi1.setTarifAwal(10000);
taksi1.setTarifPerKm(5000);
```

```
Taksi taksi2 = new Taksi();
taksi2.setJmlRoda(4);
taksi2.setWarna("Biru");
taksi2.setBahanBakar("Bensin");
taksi2.setKapasitasMesin(1300);
taksi2.setTarifAwal(7000);
taksi2.setTarifPerKm(3500);
```

```
Sepeda sepeda1 = new Sepeda();
sepeda1.setJmlRoda(3);
sepeda1.setWarna("Hitam");
sepeda1.setJmlSadel(1);
sepeda1.setJmlGir(2);
```

```
Sepeda sepeda2 = new Sepeda();
sepeda2.setJmlRoda(2);
sepeda2.setWarna("Putih");
sepeda2.setJmlSadel(2);
sepeda2.setJmlGir(5);
```

```
System.out.println("\nTruk 1");
System.out.println("jmlRoda : " + truk1.getJmlRoda());
System.out.println("warna : " + truk1.getWarna());
System.out.println("bahanBakar : " + truk1.getBahanBakar());
System.out.println("kapasitasMesin : " + truk1.getKapasitasMesin());
System.out.println("muatanMaks : " + truk1.getMuatanMaks());
System.out.println("-----");
```

```
System.out.println("\nTruk 2");
System.out.println("jmlRoda : " + truk2.getJmlRoda());
System.out.println("warna : " + truk2.getWarna());
System.out.println("bahanBakar : " + truk2.getBahanBakar());
System.out.println("kapasitasMesin : " + truk2.getKapasitasMesin());
System.out.println("muatanMaks : " + truk2.getMuatanMaks());
System.out.println("-----");
```

```
System.out.println("\nTaksi 1");
System.out.println("jmlRoda : " + taksi1.getJmlRoda());
System.out.println("warna : " + taksi1.getWarna());
```

```

System.out.println("bahanBakar : " + taksi1.getBahanBakar());
System.out.println("kapasitasMesin : " + taksi1.getKapasitasMesin());
System.out.println("tarifAwal : " + taksi1.getTarifAwal());
System.out.println("tarifPerKm : " + taksi1.getTarifPerKm());
System.out.println("-----");


System.out.println("\nTaksi 2");
System.out.println("jmlRoda : " + taksi2.getJmlRoda());
System.out.println("warna : " + taksi2.getWarna());
System.out.println("bahanBakar : " + taksi2.getBahanBakar());
System.out.println("kapasitasMesin : " + taksi2.getKapasitasMesin());
System.out.println("tarifAwal : " + taksi2.getTarifAwal());
System.out.println("tarifPerKm : " + taksi2.getTarifPerKm());
System.out.println("-----");

System.out.println("\nSepeda 1");
System.out.println("jmlRoda : " + sepeda1.getJmlRoda());
System.out.println("warna : " + sepeda1.getWarna());
System.out.println("jmlSadel : " + sepeda1.getJmlSadel());
System.out.println("jmlGir : " + sepeda1.getJmlGir());
System.out.println("-----");

System.out.println("\nSepeda 2");
System.out.println("jmlRoda : " + sepeda2.getJmlRoda());
System.out.println("warna : " + sepeda2.getWarna());
System.out.println("jmlSadel : " + sepeda2.getJmlSadel());
System.out.println("jmlGir : " + sepeda2.getJmlGir());
System.out.println("-----");
    }
}

```

g. Hasil compile

 Command Prompt

```

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>javac Tes.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>java Tes

Truk 1
jmlRoda : 4
warna : Kuning
bahanBakar : Solar
kapasitasMesin : 1500
muatanMaks : 1000
-----

Truk 2
jmlRoda : 6
warna : Merah
bahanBakar : Solar
kapasitasMesin : 2000
muatanMaks : 5000
-----

```



```
CA Command Prompt

Taksi 1
jmlRoda : 4
warna : Orange
bahanBakar : Bensin
kapasitasMesin : 1500
tarifAwal : 10000
tarifPerKm : 5000
-----

Taksi 2
jmlRoda : 4
warna : Biru
bahanBakar : Bensin
kapasitasMesin : 1300
tarifAwal : 7000
tarifPerKm : 3500
-----

Sepeda 1
jmlRoda : 3
warna : Hitam
jmlSadel : 1
jmlGir : 2
-----

Sepeda 2
jmlRoda : 2
warna : Putih
jmlSadel : 2
jmlGir : 5
-----

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 10>_
```

Analisa :

Apabila dilihat dari UML class diagram pada soal, terdapat single inheritance dan multilevel inheritance. Single inheritance ketika suatu subclass memiliki satu parent class, adapun parent class pada UML class diagram yaitu class “Kendaraan”. Multilevel inheritance ketika suatu subclass memiliki subclass lainnya seperti subclass “Mobil” memiliki subclass “Truk” dan subclass “Taksi”. Metode inheritance yang diterapkan pada soal ini yaitu setiap subclass memiliki suatu atribut yang sama yaitu “jmlRoda(int)” dan “warna(String)”, serta subclass tersebut juga memiliki metode setter dan getter yang sama terhadap atribut yang sama tersebut.